Unsupervised Learning in Python

4). Discovering interpretable features:

a). NMF applied to Wikipedia articles

```
# Import NMF
from sklearn.decomposition import NMF

# Create an NMF instance: model
model = NMF(n_components=6)

# Fit the model to articles
model.fit(articles)

# Transform the articles: nmf_features
nmf_features = model.transform(articles)

# Print the NMF features
print(nmf_features)
```

<script.py> output:

```
  [[ 0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
     0.00000000e+00   4.40501659e-01]
   [ 0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
     0.00000000e+00   5.66651628e-01]
```

b). <u>NMF features of the Wikipedia articles</u>

```
# Import pandas
import pandas as pd

# Create a pandas DataFrame: df
df = pd.DataFrame(nmf_features, index=titles)

# Print the row for 'Anne Hathaway'
print(df.loc['Anne Hathaway'])

# Print the row for 'Denzel Washington'
print(df.loc['Denzel Washington'])
```

<script.py> output:

```
0    0.003845
1    0.000000
2    0.000000
3    0.575711
4    0.000000
5    0.000000
Name: Anne Hathaway, dtype: float64
0    0.000000
1    0.005601
2    0.000000
3    0.422380
4    0.000000
5    0.000000
Name: Denzel Washington, dtype: float64
```

c). <u>NMF learns topics of documents</u>

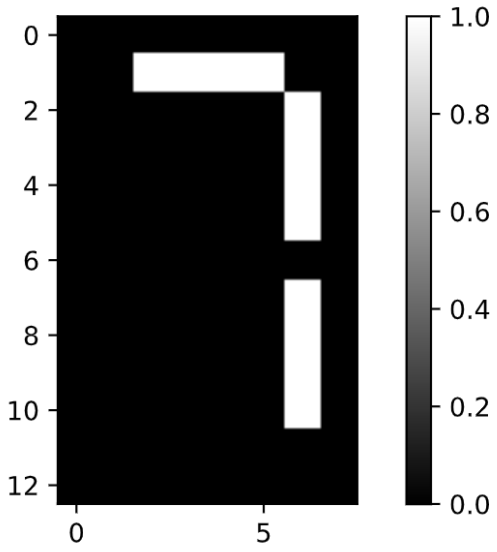
```
# Import pandas
import pandas as pd

# Create a DataFrame: components_df
components_df = pd.DataFrame(model.components_, columns=words)

# Print the shape of the DataFrame
print(components_df.shape)

# Select row 3: component
component = components_df.iloc[3,:]

# Print result of nlargest
print(component.nlargest())
```

<script.py> output:

```
(6, 13125)
film       0.627877
award      0.253131
starred    0.245284
role       0.211451
actress    0.186398
Name: 3, dtype: float64
```

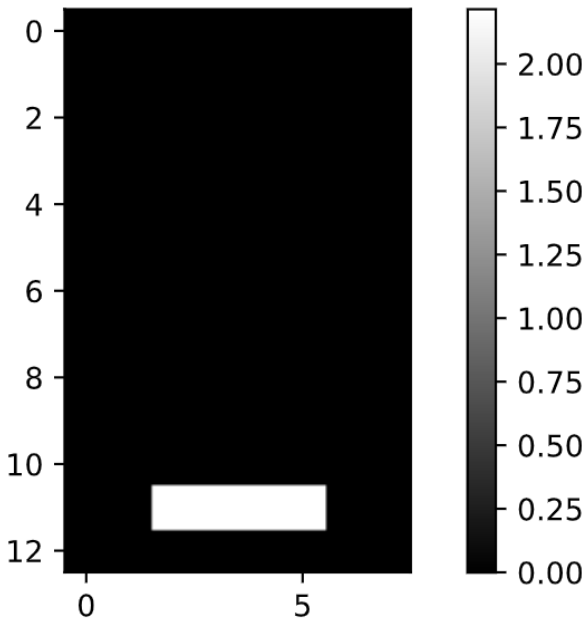
d). <u>Explore the LED digits dataset</u>

```
# Import pyplot
from matplotlib import pyplot as plt


# Select the 0th row: digit
digit = samples[0,:]


# Print digit
print(digit)


# Reshape digit to a 13x8 array: bitmap
bitmap = digit.reshape(13,8)


# Print bitmap
print(bitmap)


# Use plt.imshow to display bitmap
plt.imshow(bitmap, cmap='gray', interpolation='nearest')
plt.colorbar()
plt.show()
```



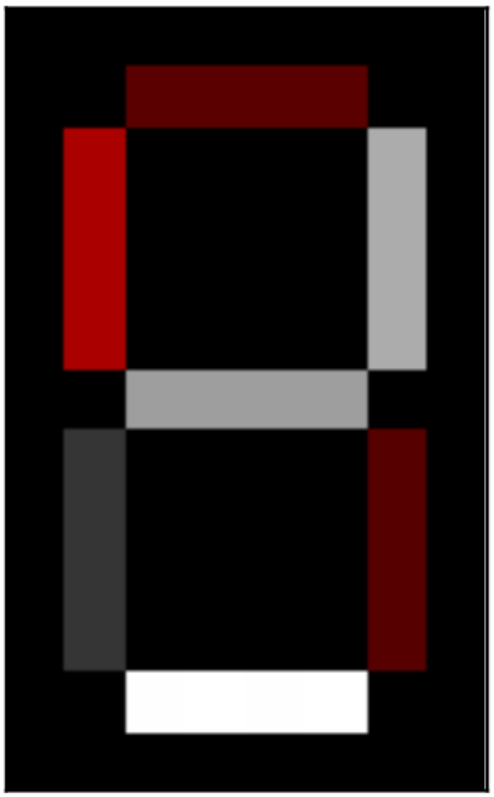
e). <u>NMF learns the parts of images</u>

```
# Import NMF
from sklearn.decomposition import NMF

# Create an NMF model: model
model = NMF(n_components=7)

# Apply fit_transform to samples: features
features = model.fit_transform(samples)

# Call show_as_image on each component
for component in model.components_:
    show_as_image(component)

# Assign the 0th row of features: digit_features
digit_features = features[0,:]

# Print digit_features
print(digit_features)
```



f). <u>PCA doesn't learn parts</u>

```
# Import PCA
from sklearn.decomposition import PCA

# Create a PCA instance: model
model = PCA(n_components=7)

# Apply fit_transform to samples: features
features = model.fit_transform(samples)

# Call show_as_image on each component
for component in model.components_:
    show_as_image(component)
```



g). <u>Which articles are similar to 'Cristiano Ronaldo'?</u>

```
# Perform the necessary imports
import pandas as pd
from sklearn.preprocessing import normalize

# Normalize the NMF features: norm_features
norm_features = normalize(nmf_features)

# Create a DataFrame: df
df = pd.DataFrame(norm_features, index=titles)

# Select the row corresponding to 'Cristiano Ronaldo': article
article = df.loc['Cristiano Ronaldo']

# Compute the dot products: similarities
similarities = df.dot(article)

# Display those with the largest cosine similarity
print(similarities.nlargest())
```

<script.py> output:

```
Cristiano Ronaldo             1.000000
Franck Ribéry                 0.999972
Radamel Falcao                0.999942
Zlatan Ibrahimović            0.999942
France national football team 0.999923
dtype: float64
```

h). <u>Recommend musical artists part I</u>

```
# Perform the necessary imports
from sklearn.preprocessing import Normalizer, MaxAbsScaler
from sklearn.pipeline import make_pipeline
from sklearn.decomposition import NMF

# Create a MaxAbsScaler: scaler
scaler = MaxAbsScaler()

# Create an NMF model: nmf
nmf = NMF(n_components=20)

# Create a Normalizer: normalizer
normalizer = Normalizer()

# Create a pipeline: pipeline
pipeline = make_pipeline(scaler, nmf, normalizer)

# Apply fit_transform to artists: norm_features
norm_features = pipeline.fit_transform(artists)
```

i). <u>Recommend musical artists part II</u>

```
# Import pandas
import pandas as pd

# Create a DataFrame: df
df = pd.DataFrame(norm_features, index=artist_names)

# Select row of 'Bruce Springsteen': artist
artist = df.loc['Bruce Springsteen']

# Compute cosine similarities: similarities
similarities = df.dot(artist)

# Display those with highest cosine similarity
print(similarities.nlargest())
```

<script.py> output:

```
Bruce Springsteen    1.000000
Neil Young           0.956153
Van Morrison         0.872528
Leonard Cohen        0.865201
Bob Dylan            0.859391
dtype: float64
```